

JUN 09 2006

Attorney's Docket No.: 07844-501001
Client's Ref. No.: P464

OFFICIAL COMMUNICATION FACSIMILE:

OFFICIAL FAX NO: (571) 273-8300

Number of pages including this page 22

Applicant : R. Perelman, et al.
Serial No. : 10/006,260
Filed : November 2, 2001

Art Unit : 2151
Examiner : Backhean Tiv

Title : CLIENT-SIDE MODIFICATION OF ELECTRONIC DOCUMENTS IN A
CLIENT-SERVER ENVIRONMENT


MAIL STOP APPEAL BRIEF - PATENTS

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Attached to this facsimile communication cover sheet is REPLY BRIEF, faxed this
9th day of June, 2006, to the United States Patent and Trademark Office.

Respectfully submitted,

Date: June 9, 2006


William E. Hunter
Reg. No. 47,671

Fish & Richardson P.C.
12390 El Camino Real
San Diego, California 92130
Telephone: (858) 678-5070
Fax: (858) 678-5099

10639656.doc

NOTE: This facsimile is intended for the addressee only and may contain privileged or confidential information. If you have received this facsimile in error, please immediately call us collect at (858) 678-5070 to arrange for its return. Thank you.

JUN 09 2006

Attorney's Docket No. 07844-501001 / P464

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant : R. Perelman, et al. Art Unit : 2151
Serial No. : 10/006,260 Examiner : Backhean Tiv
Filed : November 2, 2001
Title : CLIENT-SIDE MODIFICATION OF ELECTRONIC DOCUMENTS IN A
CLIENT-SERVER ENVIRONMENT

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

REPLY BRIEF

Pursuant to 37 C.F.R. § 41.41, Applicant responds to the Examiner's Answer mailed April 10th, 2006, referred to hereafter as Examiner's Answer, as follows:

Client Side Modification, Claims 1-40 – Examiner's Answer, reply to (a), page 18

D'Arlach (US 6,026,433) does not teach generating instructions to modify an electronic document, the generated instructions to be performed at the client. The Examiner's Answer argues that D'Arlach teaches "changes" occurring at the client since, "the page at the user is implicitly being replaced with this updated/new page, therefore there is a change that occurs at the client." See Examiner's Answer at page 18 ¶ 4. While it may be true that clients will observe the updated page, this is not a modification to the electronic document as recited in the claims. The claims recite more than merely observing a change to the document at a client. Claim 1 recites where the modification to the electronic document takes place, "...generating instructions ... specifying one more operations to modify the electronic document's predetermined final format ... the generated instructions to be performed at the client to effect the one or more operations..." See Claim 1; emphasis added. In contrast, D'Arlach discusses

CERTIFICATE OF TRANSMISSION BY FACSIMILE

I hereby certify that this correspondence is being transmitted by facsimile to the Patent and Trademark Office on the date indicated below.

Date of Transmission June 9, 2006
Signature Rita H. Jennings
Typed or Printed Name of Person Signing Certificate Rita H. Jennings

Applicant : R. Perelman, et al.
Serial No. : 10/006,260
Filed : November 2, 2001
Page : 2 of 8

Attorney's Docket No. 07844-501001 / P464

modifying the electronic document not on the client as claimed, but exclusively on the server. See D'Arlach at col. 5 lines 30-31. The D'Arlach templates reside on the server as do the generated Web pages; thus, while a user in D'Arlach may observe a "change" to the Web page, modification of the underlying electronic document only happens on the server computer. See D'Arlach at col. 6 lines 12-17.

For at least the reasons above, and the reasons stated in the Appeal Brief, all of claims 1-40 are patentable over D'Arlach, Szabo (US 6,868,525) and Dilworth (WO 00/51018), and these claims should be allowed.

Predetermined Format, Claims 1-23 and 32-40 – Examiner's Answer, reply to (b), page 19

The combination of D'Arlach with Szabo does not teach or suggest an electronic document having a predetermined format that defines an appearance of the electronic document independent of a device used to present the electronic document. The Examiner's Answer argues that, "it would have been obvious to modify the teachings of D'Arlach to use a format that will appear the same no matter what platform the computer OS is as taught by Szabo in order to display the same data on any device." See Examiner's Answer at page 20 ¶ 3; emphasis added. However, Szabo does not teach this feature.

The Examiner's Answer asserts that Szabo teaches the Extensible Markup Language ("XML") electronic document format and that the XML format is "device independent," meaning that XML "can be displayed on any OS platform." See Examiner's Answer at page 19 ¶ 2. While the statement that XML can be displayed on any platform may be true, it fails to address the claimed feature of, a "...format that defines an appearance of the electronic document independent of a device used to present the electronic document..." See Claim 1; emphasis added. The fact that XML can be displayed on any OS platform does not teach or suggest the claimed feature that the document will appear the same way on all OS platforms. XML does not provide such a feature.

In fact, the XML technologies in the references provided in the Examiner's Answer teach away from combining formatting information with XML document content. For example, the current version of Reference 3 provided in the Examiner's Answer states that, "[b]ecause XML

Applicant : R. Perelman, et al.
Serial No. : 10/006,260
Filed : November 2, 2001
Page : 3 of 8

Attorney's Docket No. 07844-501001 / P464

is a *data description* language, XML documents do not carry information about how to display the data." See <http://en.wikipedia.org/wiki/XML> at page 9; emphasis original; a copy of which is included with this response as a 13 page attachment. Reference 2 provided in the Examiner's Answer discusses delivering XML documents through various mechanisms, where the presentation or appearance of the document may vary according to the access mechanism used.¹ Reference 4 provided in the Examiner's Answer discusses XML documents mapped to different user interfaces including graphical and speech interfaces where the appearance and format of the document is adapted to a particular user interface.² Thus, while XML may be "device independent" in the sense that it is capable of being displayed on all OS platforms, XML teaches away from a format defining the document's appearance on all OS platforms as recited in the claims.

Even if XML did include formatting information, it would not define the appearance of the document independent of a device used to present it. An Hypertext Markup Language (HTML), Extensible HyperText Markup Language (XHTML), or XML document has no ability to specify which portions of the document will appear on any given physical page when printed or viewed. The display of such documents is controlled exclusively by the Web browser used to interpret and render the document and its associated formatting information. Thus, even if XML were combined with formatting information, it would fail to teach or suggest a document format that "...defines an appearance of the electronic document independent of a device used to present the electronic document..." as recited in the Claims. See Claim 1.

The Examiner's Answer cites *In re Keller* which states that, "one cannot show nonobviousness by attacking references individually where the rejections are based on combinations of references." See Examiner's Answer at page 20 ¶ 2. Applicants respectfully note that in order to establish a *prima facie* case of obviousness, the combined references

¹ "The goal is that a functional user experience should be possible via any access mechanism. The method by which presentation and interaction are provided may vary according to the different access mechanisms, but the possibility of a functional user experience should always exist." Peter Mikhaleenko, Introduction to Device Independence, Part I, September 22, 2004.

² "An objective of UIML is to permit a UIML document to be mapped to any type of user interface, from graphical to speech to those not yet invented." Marc Abrams and Constantinos Phanouriou, UIML: An XML Language for Building Device-Independent User Interfaces.

Applicant : R. Perelman, et al.
Serial No. : 10/006,260
Filed : November 2, 2001
Page : 4 of 8

Attorney's Docket No. 07844-501001 / P464

"...must teach or suggest all the claim limitations..." In re Vaeck, 947 F.2d 488 (Fed. Cir. 1991). Szabo does not teach a format "that defines an appearance of the electronic document independent of a device used to present the electronic document" as recited in the claims. D'Arlach fails to cure this deficiency of Szabo. Thus, a *prima facie* case of obviousness has not been established.

For at least the reasons above, and the reasons stated in the Appeal Brief, all of claims 1-23 and 32-40 are patentable over D'Arlach and Szabo, and these claims should be allowed.

Tags, Claims 6, 21, 24 and 37 – Examiner's Answer, reply to (c), page 21

D'Arlach does not teach generating instructions comprising at least one tag indicating an order in which the produced data is to be imported into the electronic document and the instructions are to be performed. The Examiner's Answer argues that D'Arlach teaches the recited tags implicitly: "...D'Arlach teaches that tag relates to an order in which to import data into an electronic document and to perform instructions that modify the documents to accommodate the data because D'Arlach teaches the use of HTML, which has tags and which ... indicates where elements begins and ends." See Examiner's Answer at page 21 ¶ 1. The Examiner's Answer has not made any argument to support this "implicit teaching" aside from the conclusory statement that, "HTML uses tags that indicates where elements begin and end." See Examiner's Answer at page 21 ¶ 1. While the HTML tags discussed in D'Arlach may define the beginning and ending of elements, the tags recited in the claims perform a different set of features; the recited tags, "...indicat[e] an order in which the produced data is to be imported into the electronic document and the instructions are to be performed." See Claim 6; emphasis added.

The Examiner's Answer has not shown how the HTML tags discussed in D'Arlach "indicate an order" as recited in the claims. The Examiner's Answer has not identified any "produced data" or "instructions" associated with the HTML tags discussed in D'Arlach, much less shown how the HTML tags indicate an order in which produced data is to be imported into the electronic document and instructions are to be performed. The statement that HTML tags, "indicates where elements begins and ends," does not imply the ordering feature recited in the claims. See Examiner's Answer page 21 ¶ 1. The Examiner's Answer has failed to address the

Applicant : R. Perelman, et al.
Serial No. : 10/006,260
Filed : November 2, 2001
Page : 5 of 8

Attorney's Docket No. 07844-501001 / P464

features of Claims 6, 21, 24 and 37 and, as such, the record does not support the rejection of these claims.

For at least the reasons above, and the reasons stated in the Appeal Brief, all of Claims 6, 21, 24 and 37 are patentable over D'Arlach, Szabo, and Dilworth, and these claims should be allowed.

Document Tags, Claims 7, 22, 25-27 and 38 -- Examiner's Answer, reply to (d), page 21

D'Arlach does not teach a document tag that indicates that at least a portion of generated instructions are to be inserted into an electronic document. The Examiner's Answer argues that D'Arlach implicitly teaches a "document tag" as recited in the claims because, "HTML pages uses tags ... by definition of tag, it indicates where elements begins and ends..." See Examiner's Answer page at 22 ¶ 1. The Examiner's Answer fails to address the recited claim feature; Claim 7 recites, "...the document tag indicates that at least a portion of the generated instructions are to be inserted..." See Claim 7; emphasis added. Claim 7 recites inserting generated instructions into the electronic document, not produced data as argued in the Examiner's Answer. See Examiner's Answer at page 22 ¶ 1. The Examiner's Answer has not identified any "generated instructions" associated with the HTML tags discussed in D'Arlach. Additionally, as discussed above, the HTML tags discussed in D'Arlach do not correspond to the "tags" recited in the claims. The Examiner's Answer has failed to address the features recited in the Claims 7, 22, 25-27 and 38 and, as such, the record does not support the rejection of these claims.

For at least the reasons above, and the reasons stated in the Appeal Brief, all of Claims 7, 22, 25-27 and 38 are patentable over D'Arlach, Szabo, and Dilworth, and these claims should be allowed.

Pre-Existing Document, Claims 4, 19, 29 and 35 -- Examiner's Answer, reply to (e), page 22

D'Arlach does not teach adding information to the electronic document without changing the pre-existing information in the electronic document. The Examiner's Answer asserts that in D'Arlach, "a user can add graphics, labels, and linking information to the website...not changing the other contents of the website... and add a hyperlink element for a full-text search to the

Applicant : R. Perelman, et al.
Serial No. : 10/006,260
Filed : November 2, 2001
Page : 6 of 8

Attorney's Docket No. 07844-501001 / P464

existing website ... Therefore, D'Arlach teaches adding information to the electronic document without changing the pre-existing information in the electronic document." See Examiner's Answer at page 22 ¶ 2. D'Arlach discusses templates implemented as database records containing editable elements. See D'Arlach at col. 4 lines 59-67. Thus, each storage location in the template is defined by the database schema and table before the website is edited. See D'Arlach at col. 5 lines 1-33 and Fig. 4. Accordingly, each time a template is updated in D'Arlach, the existing data in the database is overwritten, even if it that data is an empty string – this changes the pre-existing information in the document. See D'Arlach at col. 6 lines 13-16. Thus, the database storage system discussed in D'Arlach teaches away from the recited feature of, "...adding information to the electronic document without changing pre-existing information in the electronic document..." See Claim 4. The Examiner's Answer has failed to address the features recited in Claims 4, 19, 29 and 35 and, as such, the record does not support the rejection of these claims.

For at least the reasons above, and the reasons stated in the Appeal Brief, all of Claims 4, 19, 29 and 35 are patentable over D'Arlach, Szabo, and Dilworth, and these claims should be allowed.

Visual Overlay, Claims 9 and 40 – Examiner's Answer, reply to (f), page 22

D'Arlach does not teach causing a new visual object to overlay one or more pre-existing visual objects in the electronic document. The Examiner's Answer argues that Figure 10 in D'Arlach teaches this feature since, "a user can add labels, which can be a specific color, type of font, and size..." See Examiner's Answer at page 22 ¶ 3. However, the labels discussed in D'Arlach are not new visual objects, but are pre-defined in the template database. "[t]he buttons, graphics, and text ... are elements comprising a template database..." See D'Arlach at col. 8 lines 64-65 and Fig. 10. In contrast Claim 9 recites, "...wherein adding the information causes a new visual object to overlay one or more pre-existing visual objects in the electronic document." See Claim 9; emphasis added. Overlay of visual objects in an electronic document is neither described nor shown in D'Arlach and, as such, the record does not support the rejection of Claims 9 and 40.

Applicant : R. Perelman, et al.
Serial No. : 10/006,260
Filed : November 2, 2001
Page : 7 of 8

Attorney's Docket No. 07844-501001 / P464

For at least the reasons above, and the reasons stated in the Appeal Brief, Claims 9 and 40 are patentable over D'Arlach, Szabo, and Dilworth, and these claims should be allowed.

Predetermined Format, Claims 1, 16 and 32 – Examiner's Answer, reply to (h), page 23

Dilworth does not teach an electronic document format that defines an appearance of the document independent of a device used to present it. Dilworth describes an editor written in Java™ that supports “most of the HTML format, including support for graphics.” See Dilworth at Abstract. The Examiner's Answer asserts that, “it would have been obvious to modify the teachings of D'Arlach to use a format that will appear the same no matter what platform the computer OS is as taught by Dilworth.” See Examiner's Answer at Page 24 ¶ 2; emphasis added. However, Dilworth does not teach the claimed document format.

Dilworth does not purport to disclose a document format, but rather a document editor. See Dilworth at Abstract. The passage cited in the Examiner's Answer highlights this distinction: “[t]he editor supports most of the HTML formats... because the editor in the preferred embodiment is written in the Java programming language, it is cross-platform compatible with most Web browsers.” See Dilworth at page 4 lines 18-20; emphasis added. This passage clearly shows that Dilworth purports to disclose an editor of HTML formatted documents, not a document format. In contrast, the Claim 1 recites a “...predetermined format that defines an appearance of the electronic document...” See Claim 1; emphasis added. Thus, where Dilworth relies on a Java™ editor to define the appearance of an HTML formatted document, the Claims recite a document format that defines the appearance of the document. The assertion in the Examiner's Answer that, “it would have been obvious to modify the teachings of D'Arlach to use a format that will appear the same no matter what platform the computer OS is as taught by Dilworth,” cannot be supported because Dilworth does not teach a document format as claimed. See Examiner's Answer at page 24 ¶ 2.

The Examiner's Answer notes that “one cannot show nonobviousness by attacking references individually where the rejections are based on combinations of references.” In re Keller, 642 F.2d 413 (CCPA 1981). See Examiner's Answer at page 23 ¶ 4. However, Applicants respectfully note that in order to establish a *prima facie* case of obviousness, the combined references “...must teach or suggest all the claim limitations...” In re Vaerek, 947

Applicant : R. Perelman, et al.
Serial No. : 10/006,260
Filed : November 2, 2001
Page : 8 of 8

Attorney's Docket No. 07844-501001 / P464

F.2d 488 (Fed. Cir. 1991). Claim 1 recites, "...an electronic document having a predetermined format that defines an appearance of the electronic document independent of a device used to present the electronic document..." See Claim 1; emphasis added. Dilworth does not teach a document format that defines the appearance of an electronic document independent of a device used to present the document. D'Arlach does not cure this deficiency of Dilworth. Thus, the suggested combination fails to teach all of the limitations of Claims 1, 16, 24 and 32 and, as such, a *prima facie* case of obviousness has not been established.

For at least the reasons above, and the reasons stated in the Appeal Brief, all of claims 1-40 are patentable over D'Arlach and Dilworth, and these claims should be allowed

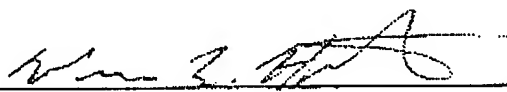
Conclusion

For these reasons, and the reasons stated in the Appeal Brief, Applicant submits that the final rejection should be reversed.

Please apply any charges or credits to Deposit Account No. 06-1050.

Respectfully submitted,

Date: June 9, 2006



William E. Hunter
Reg. No. 47,671

Fish & Richardson P.C.
12390 El Camino Real
San Diego, California 92130
Telephone: (858) 678-5070
Facsimile: (858) 678-5099

Attachment: <http://en.wikipedia.org/wiki/XML> (13 pages)

10628272.doc

XML - Wikipedia, the free encyclopedia

<http://en.wikipedia.org/wiki/XML>

XML

From Wikipedia, the free encyclopedia
(Redirected from Xml)

The **Extensible Markup Language (XML)** is a W3C-recommended general-purpose markup language for creating special-purpose markup languages, capable of describing many different kinds of data. In other words: XML is a way of describing data and an XML file can contain the data too, as in a database. It is a simplified subset of Standard Generalized Markup Language (SGML). Its primary purpose is to facilitate the sharing of data across different systems, particularly systems connected via the Internet. Languages based on XML (for example, Geography Markup Language (GML), RDF/XML, RSS, Atom, MathML, XHTML, SVG, and MusicXML) are defined in a formal way, allowing programs to modify and validate documents in these languages without prior knowledge of their form.

Contents

- 1 History
- 2 Features of XML
- 3 Strengths and weaknesses
 - 3.1 Strengths
 - 3.2 Weaknesses
- 4 Quick syntax tour
- 5 Correctness in an XML document
 - 5.1 Well-formed documents
 - 5.2 Valid documents
 - 5.2.1 DTD
 - 5.2.2 XML Schema
 - 5.2.3 RELAX NG
 - 5.2.4 Other schema languages
 - 5.3 International use
- 6 Displaying XML on the web
- 7 XML extensions
- 8 Processing XML files
- 9 Versions of XML
- 10 Patent status
- 11 See also
- 12 References
- 13 External links
 - 13.1 Specifications
 - 13.2 Basic readings
 - 13.3 Web-zines
 - 13.4 XML editors
 - 13.5 XML certification
 - 13.6 XML parsers
 - 13.7 XML tools
 - 13.8 XML mailing lists

XML - Wikipedia, the free encyclopedia

<http://en.wikipedia.org/wiki/XML>

History

The versatility of SGML for dynamic information display was understood by early digital media publishers in the late 1980s prior to the rise of the internet.^{[1][2]} By the mid-1990s some practitioners of SGML had gained experience with the then-new World Wide Web, and believed that SGML offered solutions to some of the problems the Web was likely to face as it grew. Dan Connolly added SGML to the list of W3C's activities when he joined the staff in 1995; work began in mid-1996 when Jon Bosak developed a charter and recruited collaborators. Bosak was well-connected in the small community of people who had experience both in SGML and the Web. He received support in his efforts from Microsoft.

XML was designed by an eleven-member working group, supported by an (approximately) 150-member Interest Group. Technical debate took place on the Interest Group mailing list and issues were resolved by consensus or, when that failed, majority vote of the Working Group. James Clark served as Technical Lead of the Working Group, notably contributing the empty-element "<empty/>" syntax and the name "XML". Other names that had been put forward for consideration included "MAGMA" (Minimal Architecture for Generalized Markup Applications), "SLIM" (Structured Language for Internet Markup) and "MGML" (Minimal Generalized Markup Language). The co-editors of the specification were originally Tim Bray and Michael Sperberg-McQueen. Halfway through the project Bray accepted a consulting engagement with Netscape, provoking vociferous protests from Microsoft. Bray was temporarily asked to resign the editorship. This led to intense dispute in the Working Group, eventually solved by the appointment of Microsoft's Jean Paoli as a third co-editor.

The XML Working Group never met face-to-face; the design was accomplished using a combination of email and weekly teleconferences. The major design decisions were reached in twenty weeks of intense work between July and November of 1996. Further design work continued through 1997, and XML 1.0 became a W3C Recommendation on February 10, 1998.

XML 1.0 achieved the Working Group's goals of Internet usability, general-purpose usability, SGML compatibility, facilitation of easy development of processing software, minimization of optional features, legibility, formality, conciseness, and ease of authoring.

Clarifications and minor changes were accumulated in published errata and then incorporated into a Second Edition of the XML 1.0 Recommendation on October 6, 2000. Subsequent errata were incorporated into a Third Edition on February 4, 2004.

Also published on the same day as XML 1.0 Third Edition was XML 1.1, a variant of XML that encourages more consistency in how characters are represented and relaxes restrictions on names, allowable characters, and end-of-line representations.

Both XML 1.0 Third Edition and XML 1.1 are considered current versions of XML.

Features of XML

XML provides a text-based means to describe and apply a tree-based structure to information. At its base level, all information manifests as text, interspersed with markup that indicates the information's separation into a hierarchy of *character data*, container-like *elements*, and *attributes* of those elements. In this respect, it is similar to the LISP programming language's S-expressions, which describe tree structures wherein each node may have its own property list.

The fundamental unit in XML is the *character*, as defined by the Universal Character Set. Characters are combined in certain allowable combinations to form an XML *document*. The document consists of one or more *entities*, each of which is typically some portion of the document's characters, encoded as a series of bits and stored in a text file.

XML - Wikipedia, the free encyclopedia

<http://en.wikipedia.org/wiki/XML>

The ubiquity of text file authoring software (word processors) facilitates rapid XML document authoring and maintenance, whereas prior to the advent of XML, there were very few data description languages that were general-purpose, Internet protocol-friendly, and very easy to learn and author. In fact, most data interchange formats were proprietary, special-purpose, "binary" formats (based foremost on bit sequences rather than characters) that could not be easily shared by different software applications or across different computing platforms, much less authored and maintained in common text editors.

By leaving the names, allowable hierarchy, and meanings of the elements and attributes open and definable by a customizable *schema*, XML provides a syntactic foundation for the creation of custom, XML-based markup languages. The general syntax of such languages is rigid — documents must adhere to the general rules of XML, assuring that all XML-aware software can at least read (*parse*) and understand the relative arrangement of information within them. The schema merely supplements the syntax rules with a set of constraints. Schemas typically restrict element and attribute names and their allowable containment hierarchies, such as only allowing an element named 'birthday' to contain 1 element named 'month' and 1 element named 'day', each of which has to contain only character data. The constraints in a schema may also include data type assignments that affect how information is processed; for example, the 'month' element's character data may be defined as being a month according to a particular schema language's conventions, perhaps meaning that it must not only be formatted a certain way, but also must not be processed as if it were some other type of data.

In this way, XML contrasts with HTML, which has an inflexible, single-purpose vocabulary of elements and attributes that, in general, cannot be repurposed. With XML, it is much easier to write software that accesses the document's information, since the data structures are expressed in a formal, relatively simple way.

XML makes no prohibitions on how it is used. Although XML is fundamentally text-based, software quickly emerged to abstract it into other, richer formats, largely through the use of datatype-oriented schemas and object-oriented programming paradigms (in which the document is manipulated as an object). Such software might treat XML as serialized text only when it needs to transmit data over a network, and some software doesn't even do that much. Such uses have led to "binary XML", the relaxed restrictions of XML 1.1, and other proposals that run counter to XML's original spirit and thus garner an amount of criticism.

Strengths and weaknesses

Strengths

Some features of XML that make it well-suited for data transfer are:

- its simultaneously human- and machine-readable format;
- it has support for Unicode, allowing almost any information in any human language to be communicated;
- the ability to represent the most general computer science data structures: records, lists and trees;
- the self-documenting format that describes structure and field names as well as specific values;
- the strict syntax and parsing requirements that allow the necessary parsing algorithms to remain simple, efficient, and consistent.

XML is also heavily used as a format for document storage and processing, both online and offline, and offers several benefits:

- its robust, logically-verifiable format is based on international standards;
- the hierarchical structure is suitable for most (but not all) types of documents;
- it manifests as plain text files, unencumbered by licenses or restrictions;
- it is platform-independent, thus relatively immune to changes in technology;
- it and its predecessor, SGML, have been in use since 1986, so there is extensive experience and software available.

XML - Wikipedia, the free encyclopedia

<http://en.wikipedia.org/wiki/XML>

Weaknesses

For certain applications, XML also has the following weaknesses:

- Its syntax is fairly verbose and partially redundant. This can hurt human readability and application efficiency, and yields higher storage costs. It can also make XML difficult to apply in cases where bandwidth is limited, though compression can reduce the problem in some cases. This is particularly true for multimedia applications running on cell phones and PDAs which want to use XML to describe images and video.
- Parsers should be designed to recurse arbitrarily nested data structures and must perform additional checks to detect improperly formatted or differently ordered syntax or data (this is because the markup is descriptive and partially redundant, as noted above). This causes a significant overhead for most basic uses of XML, particularly where resources may be scarce - for example in embedded systems. Furthermore, additional security considerations arise when XML input is fed from untrustworthy sources, and resource exhaustion or stack overflows are possible.
- Some consider the syntax to contain a number of obscure, unnecessary features born of its legacy of SGML compatibility. However, an effort to settle on a subset called "Minimal XML" led to the discovery that there was no consensus on *which* features were in fact obscure or unnecessary.
- The basic parsing requirements do not support a very wide array of data types, so interpretation sometimes involves additional work in order to process the desired data from a document. For example, there is no provision in XML for mandating that "3.14159" is a floating-point number rather than a seven-character string. Some XML schema languages add this functionality.
- Uses the hierarchical model for representation, which is limited compared to the relational model, since it only gives a fixed view of the actual information. For example either actors under movies, or movies under actors.
- Modelling overlapping (non-hierarchical) data structures requires extra effort.
- Mapping XML to the relational or object oriented paradigms is often cumbersome.
- Some have argued that XML can be used for data storage only if the file is of low volume, but this is only true given particular assumptions about architecture, data, implementation, and other issues.

See also: "XML Sucks" on c2.com (<http://c2.com/cgi/wiki?XmlSucks>)

Quick syntax tour

The basic syntax for one element in XML is

```
<name attribute="value">content</name>
```

Here is an example of a simple recipe expressed using XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<recipe name="bread" prep_time="5 mins" cook_time="3 hours">
  <title>Basic bread</title>
  <ingredient amount="1" unit="cups">Flour</ingredient>
  <ingredient amount="0.25" unit="ounce">Yeast</ingredient>
  <ingredient amount="1.5" unit="cups" state="warm">Water</ingredient>
  <ingredient amount="1" unit="teaspoon">Salt</ingredient>
  <instructions>
    <step>Mix all ingredients together, and knead thoroughly.</step>
    <step>Cover with a cloth, and leave for one hour in warm room.</step>
    <step>Knead again, place in a tin, and then bake in the oven.</step>
  </instructions>
</recipe>
```

The first line is the **XML declaration**: it is an optional line stating what version of XML is in use (normally version 1.0), and may also contain information about character encoding and external dependencies.

XML - Wikipedia, the free encyclopedia

<http://en.wikipedia.org/wiki/XML>

The remainder of this document consists of nested *elements*, some of which have *attributes* and *content*. An **element** typically consists of two tags, a *start tag* and an *end tag*, possibly surrounding text and other elements. The **start tag** consists of a name surrounded by angle brackets, like "`<step>`"; the **end tag** consists of the same name surrounded by angle brackets, but with a forward slash preceding the name, like "`</step>`". The element's **content** is everything that appears between the start tag and the end tag, including text and other (child) elements. The following is a complete XML element, with start tag, text content, and end tag:

```
<step>Knead again, place in a tin, and then bake in the oven.</step>
```

In addition to content, an element can contain **attributes** — name-value pairs included in the start tag after the element name. Attribute values must always be quoted, using single or double quotes, and each attribute name should appear only once in any element.

```
<ingredient amount="3" unit="cups">Flour</ingredient>
```

In this example, the *ingredient* element has two attributes: *amount*, having value "3", and *unit*, having value "cups". In both cases, at the markup level, the names and values of the attributes, just like the names and content of the elements, are just textual data — the "3" and "cups" are not a quantity and unit of measure, respectively, but rather are just character sequences that the document author may be using to *represent* those things.

In addition to text, elements may contain other elements:

```
<instructions>
  <step>Mix all ingredients together, and knead thoroughly.</step>
  <step>Cover with a cloth, and leave for one hour in warm room.</step>
  <step>Knead again, place in a tin, and then bake in the oven.</step>
</instructions>
```

In this case, the *instructions* element contains three *step* elements. XML requires that elements be properly nested — elements may never overlap. For example, this is not well-formed XML, because the *em* and *strong* elements overlap:

```
<!-- WRONG! NOT WELL-FORMED XML! -->
<p>Normal ememstrong emphasized </em> strongstrong</strong></p>
```

Every XML document must have exactly one top-level **root element** (alternatively called a *document element*), so the following would also be a malformed XML document:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- WRONG! NOT WELL-FORMED XML! -->
<thing>Thing one</thing>
<thing>Thing two</thing>
```

XML provides special syntax for representing an element with empty content. Instead of writing a start tag followed immediately by an end tag, a document may contain the **empty element tag** where a slash *follows* the element name. The following two examples are functionally equivalent:

```
<foo></foo>
```

XML - Wikipedia, the free encyclopedia

<http://en.wikipedia.org/wiki/XML>

```
<foo />
```

XML provides two methods for escaping (or simply representing) special characters: *entity references* and *numeric character references*.

An **entity** in XML is a named body of data, usually text, such as an unusual character.

An **entity reference** is a placeholder that represents that entity. It consists of the entity's name preceded by an ampersand ("&") and followed by a semicolon (";"). XML has five predeclared entities:

- & (&)
- < (<)
- > (>)
- ' (')
- " (")

Here is an example using a predeclared XML entity to represent the ampersand in the name "AT&T":

```
<company-name>AT&amp;T</company-name>
```

If more entities need to be declared, this is done in the document's DTD. A basic example of doing so in a minimal internal DTD follows. Declared entities can describe single characters or pieces of text, and can reference each other.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE example [
  <ENTITY copy "&xA9;">
  <ENTITY copyright-notice "Copyright &copy; 2006, XYZ Enterprises">
]>
<root>
  &copyright-notice;
</root>
```

When viewed in a suitable browser, such as Firefox, the XML document above appears as:

```
<root> Copyright © 2006, XYZ Enterprises </root>
```

Numeric character references look like entities, but instead of a name, they contain the "&#x" character followed by a number. The number (in decimal or "x"-prefixed hexadecimal) represents a Unicode code point, and is typically used to represent characters that are not easily encodable, such as an Arabic character in a document produced on a European computer. The ampersand in the "AT&T" example could also be escaped like this (decimal 38 and hexadecimal 26 both represent the Unicode value for the "&" character):

```
<company-name>AT&#38;T</company-name>
<company-name>AT&#x26;T</company-name>
```

There are many more rules necessary to be sure of writing well-formed XML documents, such as the exact characters allowed in an XML name, but this quick tour provides the basics necessary to read and understand many XML documents.

XML - Wikipedia, the free encyclopedia

<http://en.wikipedia.org/wiki/XML>

Correctness in an XML document

For an XML document to be correct, it must be:

- **Well-formed.** A well-formed document conforms to all of XML's syntax rules. For example, if a non-empty element has an opening tag with no closing tag, it is not *well-formed*. A document that is not well-formed is not considered to be XML; a parser is required to refuse to process it.
- **Valid.** A valid document has data that conforms to a particular set of user-defined content rules, or XML schemas, that describe correct data values and locations. For example, if an element in a document is required to contain text that can be interpreted as being an integer numeric value, and it instead has the text "hello", is empty, or has other elements in its content, then the document is not *valid*.

Well-formed documents

An XML document is text, which is a sequence of characters. The specification requires support for Unicode encodings UTF-8 and UTF-16 (UTF-32 is not mandatory). The use of other non-Unicode based encodings, such as ISO-8859, is admitted and is indeed widely used and supported.

A well-formed document must conform to the following rules, among others:

- One and only one root element exists for the document. However, the XML declaration, processing instructions, and comments can precede the root element.
- Non-empty elements are delimited by both a start-tag and an end-tag.
- Empty elements may be marked with an empty-element (self-closing) tag, such as `<IAmEmpty />`. This is equal to `<IAmEmpty></IAmEmpty>`.
- All attribute values are quoted, either single (') or double (") quotes. Single quotes close a single quote and double quotes close a double quote.
- Tags may be nested but must not overlap. Each non-root element must be completely contained in another element.
- The document complies to its character set definition. The charset is usually defined in the xml declaration but it can be provided by the transport protocol, such as HTTP. If no charset is defined, usage of a Unicode encoding is assumed, defined by the Unicode Byte Order Mark. If the mark does not exist, UTF-8 is the default.

Element names are case-sensitive. For example, the following is a well-formed matching pair

```
<Step> ... </Step>
```

whereas this is not

```
<Step> ... </step>
```

The careful choice of names for XML elements will convey the meaning of the data in the markup. This increases human readability while retaining the rigor needed for software parsing.

Choosing meaningful names implies the semantics of elements and attributes to a human reader without reference to external documentation. However, this can lead to verbosity, which complicates authoring and increases file size.

Valid documents

An XML document that complies with a particular schema, in addition to being well-formed, is said to be **valid**.

XML - Wikipedia, the free encyclopedia

<http://en.wikipedia.org/wiki/XML>

An XML schema is a description of a type of XML document, typically expressed in terms of constraints on the structure and content of documents of that type, above and beyond the basic constraints imposed by XML itself. A number of standard and proprietary XML schema languages have emerged for the purpose of formally expressing such schemas, and some of these languages are XML-based, themselves.

Before the advent of generalised data description languages such as SGML and XML, software designers had to define special file formats or small languages to share data between programs. This required writing detailed specifications and special-purpose parsers and writers.

XML's regular structure and strict parsing rules allow software designers to leave parsing to standard tools, and since XML provides a general, data model-oriented framework for the development of application-specific languages, software designers need only concentrate on the development of rules for their data, at relatively high levels of abstraction.

Well-tested tools exist to validate an XML document "against" a schema: the tool automatically verifies whether the document conforms to constraints expressed in the schema. Some of these validation tools are included in XML parsers, and some are packaged separately.

Other usages of schemas exist: XML editors, for instance, can use schemas to support the editing process.

DTD

The oldest schema format for XML is the Document Type Definition (DTD), inherited from SGML. While DTD support is ubiquitous due to its inclusion in the XML 1.0 standard, it is seen as limited for the following reasons:

- It has no support for newer features of XML, most importantly namespaces.
- It lacks expressivity. Certain formal aspects of an XML document cannot be captured in a DTD.
- It uses a custom non-XML syntax, inherited from SGML, to describe the schema.

XML Schema

A newer XML schema language, described by the W3C as the successor of DTDs, is XML Schema, or more informally referred to in terms of the initialism for XML Schema instances, XSD (XML Schema Definition). XSDs are far more powerful than DTDs in describing XML languages. They use a rich datatyping system, allow for more detailed constraints on an XML document's logical structure, and are required to be processed in a more robust validation framework. Additionally, XSDs use an XML-based format, which makes it possible to use ordinary XML tools to help process them, although WXS (W3C XML Schema) implementations require much more than just the ability to read XML.

Criticisms of WXS include the following:

- The specification is very large, which makes it difficult to understand and implement.
- The XML-based syntax leads to verbosity in schema description, which makes XSDs harder to read and write.

RELAX NG

Another popular schema language for XML is RELAX NG. Initially specified by OASIS, RELAX NG is now also an ISO international standard (as part of DSDI). It has two formats: an XML-based syntax and a non-XML compact syntax. The compact syntax aims to increase readability and writability, but since there is a well-defined way to translate compact syntax to the XML syntax and back again by means of James Clark's Trang conversion tool (<http://www.thaiopensource.com/relaxng/trang.html>), the advantage of using standard XML tools is not lost.

XML - Wikipedia, the free encyclopedia

<http://en.wikipedia.org/wiki/XML>

Compared to XML Schema, RELAX NG has a simpler definition and validation framework, making it easier to use and implement. It also has the ability to use any datatype framework on a plug-in basis; for example, a RELAX NG schema author can require values in an XML document to conform to definitions in XML Schema Datatypes.

Other schema languages

Some schema languages not only describe the structure of a particular XML format but also offer limited facilities to influence processing of individual XML files that conform to this format. DTDs and XSDs both have this ability; they can for instance provide attribute defaults. RELAX NG intentionally does not provide these facilities.

International use

XML supports the direct use of almost any Unicode character in element names, attributes, comments, character data, and processing instructions. Therefore, the following is a well-formed XML document, even though it includes both Chinese and Russian characters:

```
<?xml version="1.0" encoding="UTF-8"?>
<? ? 2777777?/? ? >
```

Displaying XML on the web

Because XML is a *data description* language, XML documents do not carry information about how to display the data. Without using CSS or XSL, a generic XML document is rendered as raw XML text by most web browsers. Some display it with 'handles' (e.g. + and - signs in the margin) that allow parts of the structure to be expanded or collapsed with mouse-clicks.

In order to style the rendering in a browser with CSS, the XML document must include a reference to the stylesheet:

```
<?xml-stylesheet type="text/css" href="myStyleSheet.css"?>
```

See the CSS article for an example of this in action.

Note that this is different from specifying such a stylesheet in HTML, which uses the `<link>` element.

Extensible Stylesheet Language (XSL) can be used to alter the format of XML data, either into HTML or other formats that are suitable for a browser to display.

To specify client-side XSL Transformation (XSLT), the following processing instruction is required in the XML:

```
<?xml-stylesheet type="text/xsl" href="myTransform.xslt"?>
```

Client-side XSLT is supported by many web browsers, but not Opera before version 9.0. An alternative, rather than be dependent on the end-user's browser capabilities, is to use XSL to convert XML into a displayable format *on the server*. The end-user is not aware of what has gone on 'behind the scenes', all they see is well-formatted, displayable data.

See the XSLT article for an example of server-side XSLT in action.

XML extensions

XML - Wikipedia, the free encyclopedia

<http://en.wikipedia.org/wiki/XML>

- **XPath** makes it possible to refer to individual parts of an XML document. This provides random access to XML data for other technologies, including XSLT, XSL-FO, XQuery etc. XPath expressions can refer to all or part of the text, data and values in XML elements, attributes, processing instructions, comments etc. They can also access the names of elements and attributes. XPaths can be used in both valid and well-formed XML, with and without defined namespaces.
- **XQuery** is to XML what SQL is to relational databases.
- **XML namespaces** enable the same document to contain XML elements and attributes taken from different vocabularies, without any naming collisions occurring.
- **XML Signature** defines the syntax and processing rules for creating digital signatures on XML content.
- **XML Encryption** defines the syntax and processing rules for encrypting XML content.
- **XPointer** is a system for addressing components of XML-based internet media.

Processing XML files

SAX and DOM are object oriented programming APIs widely used to process XML data. The first XML parsers exposed the contents of XML documents to applications as SAX events or DOM objects.

SAX is a lexical, event-driven interface in which a document is read serially and its contents are reported as "callbacks" to various methods on a handler object of the user's design. SAX is fast and efficient to implement, but difficult to use for extracting information at random from the XML, since it tends to burden the application author with keeping track of what part of the document is being processed. It is better suited to situations in which certain types of information are always handled the same way, no matter where they occur in the document.

DOM is an interface-oriented API that allows for navigation of the entire document as if it were a tree of "Node" objects representing the document's contents. A DOM document can be created by a parser, or can be generated manually by users (with limitations). Data types in DOM Nodes are abstract; implementations provide their own programming language-specific bindings. DOM implementations tend to be memory intensive, as they generally require the entire document to be loaded into memory and constructed as a tree of objects before access is allowed.

Another form of XML Processing API is data binding, where XML data is made available as a custom, strongly typed programming language data structure, in contrast to the interface-oriented DOM. Example data binding systems are the Java Architecture for XML Binding (JAXB) [1] (<http://java.sun.com/xml/jaxb/>) and the Strathclyde Novel Architecture for Querying XML (SNAQue) [2] (<http://www.cis.strath.ac.uk/research/snaque/>).

A filter in the Extensible Stylesheet Language (XSL) family can transform an XML file for displaying or printing.

- **XSL-FO** is a declarative, XML-based page layout language. An XSL-FO processor can be used to convert an XSL-FO document into another non-XML format, such as PDF.
- **XSLT** is a declarative, XML-based document transformation language. An XSLT processor can use an XSLT *stylesheet* as a guide for the conversion of the data tree represented by one XML document into another tree that can then be serialized as XML, HTML, plain text, or any other format supported by the processor.
- **XQuery** is a W3C language for querying, constructing and transforming XML data.
- **XPath** is a DOM-like node tree data model and path expression language for selecting data within XML documents. XSL-FO, XSLT and XQuery all make use of XPath. XPath also includes a useful function library.

The native file format of OpenOffice.org, AbiWord, and Apple's iWork applications is XML. Some parts of Microsoft Office 11 will also be able to edit XML files with a user-supplied schema (but not a DTD), and on June 2, 2005 Microsoft announced that, by late 2006 all the files created by users of its Office suite of software will be formatted with web-centered XML specifications. There are dozens of other XML editors available.

XML - Wikipedia, the free encyclopedia

<http://en.wikipedia.org/wiki/XML>

Versions of XML

There are two current versions of XML. The first, *XML 1.0*, was initially defined in 1998. It has undergone minor revisions since then, without being given a new version number, and is currently in its third edition, as published on February 4, 2004. It is widely implemented and still recommended for general use. The second, *XML 1.1*, was initially published on the same day as XML 1.0 Third Edition. It contains features — some contentious — that are intended to make XML easier to use for certain classes of users (mainframe programmers, mainly). XML 1.1 is not very widely implemented and is recommended for use only by those who need its unique features.

XML 1.0 and XML 1.1 differ in the requirements of characters used for element and attribute names: XML 1.0 only allows characters which are defined in Unicode 2.0, which includes most world scripts, but excludes those which were added in later Unicode versions. Among the excluded scripts are Mongolian, Cambodian, Amharic, Burmese, and others.

Almost any Unicode character can be used in the character data and attribute values of an XML 1.1 document, even if the character is not defined, aside from having a code point, in the current version of Unicode. The approach in XML 1.1 is that only certain characters are forbidden, and everything else is allowed, whereas in XML 1.0, only certain characters are explicitly allowed, thus XML 1.0 cannot accommodate the addition of characters in future versions of Unicode.

In character data and attribute values, XML 1.1 allows the use of more control characters than XML 1.0, but, for "robustness", most of the control characters introduced in XML 1.1 must be expressed as numeric character references. Among the supported control characters in XML 1.1 are two line break codes that must be treated as whitespace. Whitespace characters are the only control codes that can be written directly.

There are also discussions on an XML 2.0, although it remains to be seen if such will ever come about. XML-SW (SW for skunk works), written by one of the original developers of XML, contains some proposals for what an XML 2.0 might look like: elimination of DTDs from syntax, integration of namespaces, XML Base and XML Information Set (*infoset*) into the base standard.

The World Wide Web Consortium also has a XML Binary Characterization Working Group doing preliminary research into use cases and properties for a binary encoding of the XML infoset. The working group is not chartered to produce any official standards. Since XML is by definition text-based, ITU-T and ISO are using the name *Fast Infoset* (<http://asn1.elibel.tm.fr/xml/finf.htm>) for their own binary infoset to avoid confusion (see ITU-T Rec. X.891 | ISO/IEC 24824-1).

Patent status

In October 2005 the small company Scientigo publicly asserted that two of its patents, U.S. Patent 5842213 (<http://patft.uspto.gov/netacgi/nph-Parser?patentnumber=5842213>) and U.S. Patent 6393426 (<http://patft.uspto.gov/netacgi/nph-Parser?patentnumber=6393426>), apply to the use of XML. The patents cover the transfer of "data in neutral forms", according to their applications, which were filed in 1997 and 1999. Scientigo CEO Doyal Bryant expressed a desire to "monetize" the patents but stated that the company was "not interested in having us against the world." He said that Scientigo was discussing the patents with several large corporations.[3] (http://news.com.com/Small+company+makes+big+claims+on+XML+patents/2100-1014_3-5905949.html)

XML users and independent experts responded to Scientigo's claims with widespread skepticism and criticism. Some derided the company as a patent troll. Tim Bray described any claims that the patents covered XML as "ridiculous on the face of it"[4] (<http://blogs.zdnet.com/BTL/?p=2052>). Because there exists a large amount of prior art relating to XML, some legal experts believed it would be difficult for Scientigo to enforce its patents through litigation.

XML - Wikipedia, the free encyclopedia

<http://en.wikipedia.org/wiki/XML>

Microsoft has also been accused of applying for patents that, if granted, could restrict the use of XML.[5]
(http://news.com.com/2100-1013_3-5146581.html)

See also

- CDATA section, the mechanism for including non-markup text in XML
- XML schema languages: DTD, XML Schema, RELAX NG, REXL
- XML processing APIs: DOM, SAX, EXSLT
- XML Protocol
- DocBook
- DocBook XML
- XSL
- S-expression, XML query language
- XRI, XDI
- Extensible Metadata Platform (XMP), used in graphics applications
- ASN.1
- WBXML
- Serialization
- List of general purpose markup languages
- Comparison of layout engines (XML)
- Single source publishing
- Extensible Binary Meta Language
- YAML
- JSON
- XML Data Binding

References

1. ^ Bray, Tim (February 2005). A conversation with Tim Bray: Searching for ways to tame the world's vast stores of information (<http://www.acmqueue.com/modules.php?name=Content&pa=showpage&pid=282>) . Association for Computing Machinery's "Queue site". URL accessed on April 16, 2006.
2. ^ (1988) "Publishers, multimedia, and interactivity", *Interactive multimedia* (http://www.amazon.com/gp/product/1556151241/qid=1145565392/sr=1-1/ref=sr_1_1/002-9630911-4966) . Cobb Group. ISBN 1556151241.

External links

Specifications

- W3C XML homepage (<http://www.w3.org/XML/>)
- The XML 1.0 specification (<http://www.w3.org/TR/REC-xml>)
- The XML 1.1 specification (<http://www.w3.org/TR/xml11>)



Wikibooks has more about this subject:
XML

Basic readings

- The XML FAQ (<http://xml.silmaril.ie/>)
- Annotated XML Specification (<http://www.xml.com/axml/testxml.htm>)
- Absolute Beginners' XML Tutorial (http://www.mousewhisperer.co.uk/xml_page.html)

XML - Wikipedia, the free encyclopedia

<http://en.wikipedia.org/wiki/XML>

- XML Tutorial for Beginners (<http://www.academictutorials.com/xml/>)
- W3 Schools - Learn XML (<http://www.w3schools.com/xml/>)
- Introduction to XML (<http://articles.pc-news.org/introduction-to-xml/computers-and-technology>)
- XML: Some hyperlinks minus the hype (<http://homepages.inf.ed.ac.uk/wadler/xml/>) by Philip Wadler

Web-zines

- XML.com (<http://www.xml.com/>)

XML editors

See *XML editor*.

XML certification

- JavaRanch XML Certification Discussion Forum (<http://saloon.javaranch.com/cgi-bin/ubb/ultimatebb.cgi?ubb=forum&f=52>)
- XML Certification (<http://www.whizlabs.com/products/xmlwhiz/xmlwhiz.html>)

XML parsers

- Microsoft XML Parser (MSXML) specification (<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/xmlsdk/html/b24aafc2-bfb1-b-4702-bf1c-b7a>)
- Expat XML Parser in C in SourceForge.net (<http://expat.sourceforge.net/>)
- Apache Xerces (<http://xerces.apache.org/>)
- DOM4J XML Parser for Java (<http://dom4j.org/>)
- CookXml XML Parser for Java (<http://cookxml.sourceforge.net/>)
- TinyXml Parser for C++ programmers (<http://www.grinninglizard.com/tinyxml/>)
- XimpleWare's VTD-XML XML Parser in both C and Java (<http://vtd-xml.sf.net>)
- Stefan Heymann's free XML Parser for Delphi (<http://www.destructor.de>)

XML tools

- Xml and tools for Xml or in Xml (<http://www.xul.fr/en-xml.html>)
- Online XML *pretty printer* (<http://www.iconv.com/xmllint.htm>) Improve human readability of XML documents.

XML mailing lists

- XML-DEV Mailing List (<http://www.xml-dev.com:7070/list/xmldev.en.html>)

Retrieved from "<http://en.wikipedia.org/wiki/XML>"

Categories: XML | Markup languages | W3C standards | Technical communication | Computer file formats | Abbreviations | Data modelling languages

-
- This page was last modified 20:40, 1 June 2006.
 - All text is available under the terms of the GNU Free Documentation License (see **Copyrights** for details). Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc.